Practical Experiences with Chronics Discovery in Large Telecommunications Systems *

Soila P. Kavulya†, Kaustubh Joshi§, Matti Hiltunen§, Scott Daniels§,
Rajeev Gandhi†,Priya Narasimhan†
Carnegie Mellon University†
{spertet,rgandhi,priyan}@ece.cmu.edu

AT&T Labs - Research
{kaustubh,hiltunen,daniels}@research.att.com§

ABSTRACT

Chronics are recurrent problems that fly under the radar of operations teams because they do not perturb the system enough to set off alarms or violate service-level objectives. The discovery and diagnosis of never-before seen chronics poses new challenges as they are not detected by traditional threshold-based techniques, and many chronics can be present in a system at once, all starting and ending at different times. In this paper, we describe our experiences diagnosing chronics using server logs on a large telecommunications service. Our technique uses a scalable Bayesian distribution learner coupled with an information-theoretic measure of distance (KL divergence), to identify the attributes that best distinguish failed calls from successful calls. Our preliminary results demonstrate the usefulness of our technique by providing examples of actual instances where we helped operators discover and diagnose chronics.

1. INTRODUCTION

Chronics are the low-grade fevers of large distributed services that support millions of users. These problems fly under the radar of operations teams because they are not big enough to set off alarm thresholds, yet they result in significant degradation in user satisfaction if left unresolved over long periods of time. Chronics can be recurrent, occurring repeatedly but unpredictably for short durations of time, or they may persist, affecting small subsets of users all the time.

The discovery and diagnosis of never-before seen chronics is a task that poses new challenges compared to the diagnosis of sudden system outages. Threshold-based detection techniques [4, 6, 13] do not work well because lowering the thresholds to detect chronics would increase the number of false positives. In addition, they pose challenges to other techniques too. Chronics that occur over short periods of time often go away before diagnosis can be performed, while those that persist for long periods of time can get absorbed into a system's definition of "normal", thus creating problems for change-point detection methods [1] or those that rely on historical models [9]. Unlike major outages which are rare and often have a single cause, lots of chronics can be present in a system at once, all starting and ending at different times. Furthermore, they may also be hidden by larger, more pressing problems. These characteristics make it difficult to isolate and diagnose individual chronic problems or isolate periods of bad system behavior to narrow regions of time that can be analyzed in more detail [17].

In addition to these unique challenges, the traditional problems of large systems diagnosis remain for chronics as well. Service platforms often consist of hundreds (or more) of network and server elements of different types, from different vendors, often producing high volumes of partially-structured logs with different formats. Detecting problems based on low-level events such as resource utilization indicators or network packet loss does not translate directly to user-visible symptoms resulting in false positives, while waiting for customer complaints to identify user-problems is often too late. Finally, a variety of different underlying problems may cause user requests to fail—failures may be caused by the service elements (*e.g.*, due to upgrades or misconfiguration), the underlying network, customer issues (*e.g.*, misconfiguration or misuse), or combinations of the above.

In this paper, we report on our experiences addressing these issues to detect chronics using server logs on a large production telecommunications service that handles tens of millions of calls per day. Specifically, we analyze call detail record (CDR) logs collected from a part of a major US ISP's Voice over IP telephone net-

*Reprinted from the proceedings of SLAML'11 with permission.

work over a period of several months. VoIP is an application with rapidly increasing importance. The user base of VoIP will increase to hundreds of millions of users [18] by replacing traditional telephony in the offerings of many wireline ISPs (*e.g.*, [7,15]), and in upcoming 4G cellular standards such as LTE that require the use of VoIP for all cellular telephony [18]. However, we will argue that the problems we address, and our solutions, are not limited to VoIP; they are likely to be applicable to many other large systems such as Internet services that are used by millions of users as well.

After discussing the suitability and limitations of existing diagnosis literature to address the above challenges, we propose the outline of a new statistical approach for chronic discovery and diagnosis that is scalable, yet does not use historical data for learning the "normal" behavior of the system. Therefore, it is ideal for discovering unexpected problems that have never been seen before, and it can also identify problems that have persisted in the background for a long time. It is designed to isolate independent problems that may be ongoing at the same time, and even on common network elements. Finally, with a little domain-specific effort, it can work with semi-structured system logs from different vendors with different structures and semantics.

The methodology we propose is simple: a) we start from the top and label every user-interaction with the system as normal or anomalous using simple application heuristics, b) using application-specific keys, we then correlate as much system log data to each labeled interaction as is possible, and c) we finally run a scalable Bayesian distribution learner coupled with an information-theoretic measure of distance (KL divergence [11]), to identify groups of call attributes in the correlated dataset that best distinguish failed user interactions from successful ones. Finally, we show that the technique is powerful—preliminary applications of the technique to our target VoIP service have helped discover a number of actual chronics.

2. THE DATASET AND CHALLENGES

We investigate chronics discovery for a part of the `VoIP` operations of a major US-based ISP. However, our findings may also apply to other large distributed systems, (*e.g.*, e-commerce, web-search, social networks) that serve users via independent interactions such as web requests. The portion of the ISP's `VoIP` network that we analyzed handles tens of millions of calls each day, contains several hundred network elements, and is layered on a large IP backbone. The network offers a portfolio of voice services including individual accounts, teleconferencing, self-managed solutions where customers manage their own premise equipment (PBXs), and wholesale customers who buy network minutes in bulk and resell them. Each service has different call path patterns, with calls going through combinations of network elements such as VoIP gateways (IPBEs), traditional phone gateways (GSXs), accounting servers, application servers (AS), voicemail servers, and policy servers (PSX). Many of these are built by different vendors and have different log file formats.

Table 1: A Generic Call Detail Record (CDR).

Attribute	Description
Timestamps	Call start and end times
Service	Type of service
Caller/callee info	Phone number and IP address
Network Element	Name of network element, *e.g.*, gateway X
Defect code	Problem encountered, *e.g.*, server timeout

To satisfy the high availability requirements of the system, there are real-time operations teams that monitor both low-level alarms derived from the equipment (server and network errors, CPU/memory/network utilization counters, *etc.*), as well as end-to-end indicators such as customer complaints and output from automated test call systems. Codebook-based systems [19] that are driven by signatures of known problems are used for identifying related alarms and for diagnosis. Major outages often result in immediate impact on successful call volumes, alarms from many sources, and are usually detected and resolved quickly.

Despite such robust operations support, the system always has a number of call defects occurring at any time of the day in the form of "background noise". The causes are many, ranging from network elements that need to be reset or rebooted, to protocol compatibility issues for corner cases, to configuration problems for individual customers. Measured in defects per million (DPM), they represent only a small fraction of the calls at any given time, but left unchecked, they can add up quickly over weeks and months. A separate chronics team troubleshoots these defects, but diagnosis is still a largely manual process. We seek to provide tools that can help such chronics teams quickly discover low-grade problems that are hidden in the background noise.

2.1 Call Detail Logs

We examined logs from the `VoIP` network over a period of several months. An example is shown below. These include call detail record (CDR) logs that are generated locally by many network elements for each call that passes through them. The logs often contain hundreds of fields that specify many details of the call such as the caller and callee information, local resources and circuits used by the call, call start and end time, and error codes, if any. The structure and semantics of these records are vendor-specific. However, many logs include several common fields, some of which are shown in Table 1. However, even the fields that are common may not always match - *e.g.*, some servers record all the

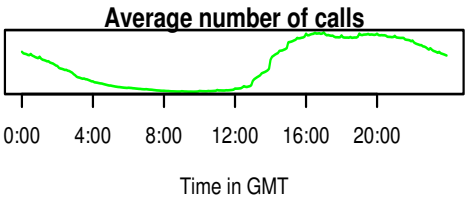

Figure 1: System usage follows cyclical patterns. Call counts were obscured to preserve privacy.

digits in the calling and called phone numbers, while others omit the last four digits. These logs tend to be large—the average size of the raw CDR logs is 30GB/-day. Even after significant consolidation to eliminate irrelevant data fields, the average size is 2.4GB/day, and each log contains between 1500-3000 unique call attributes pertinent to diagnosis.

```
# CDR Log snippet
20100901064914,STOP,at4ga01gh,phnum1,phnum2,
PSTN-TO-IP,ipaddr1:32620/ipaddr2:25378,
CCE_PHILAPASLCA3_TG,
otg=0001ATLNGANW05T-T0012,
VENDOR_X_CCE,Answered,Success,ph4pa0102sce
```

2.2 Challenges in Diagnosing Chronics

Chronics occur for a variety of reasons. For example, customer misconfiguration affecting some, but not all, calls made by the customer. These failures persist until the customer fixes their configuration. Increases in system workload can also cause regularly occurring chronics (*e.g.*, peak business hours). Such problems may be due to under-provisioning or customer exceeding their resource caps (*e.g.*, number of concurrent calls). Equipment failures such as a bad row of memory can also cause random call failures, but at a rate lower than would trigger an alarm. Operators could ignore these problems if they were one-off incidents. However, the recurrent nature of these problems negatively impacts customer satisfaction over time. Diagnosing chronics poses the following challenges.

Chronics fly under the radar.

Chronics typically occur sporadically, or affect a small subset customers, and thus do not trigger any threshold-based alarms. This is because setting threshold conditions is notoriously difficult. As shown in Figure 1, large systems often have cyclical fluctuations in volume over the course of a day, days of the week, and even month of the year. This makes it difficult to set static thresholds for what is normal. Techniques to learn historical trends and thus change thresholds dynamically exist, but chronics often live in the small gaps between predicted trends and actual values.

Multiple independent problems.

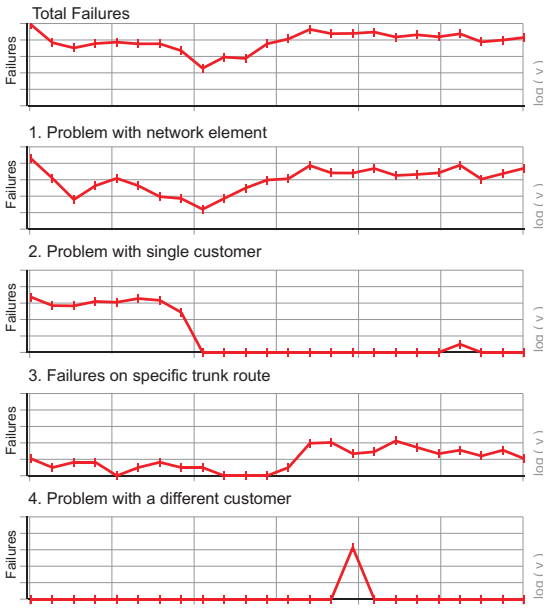

Figure 2: Defects associated with a network element may be due to many different causes.

Because chronics often persist for long periods of time before they are discovered, there are usually many of them ongoing at the same time. For example, Figure 2 shows an actual example where the total number of defective calls passing through a network element (top graph) over a period of 3 days were in fact due to atleast four unique problems - one related to a network element, two related to two different customers (problems 2 and 4), and one a capacity problem with a trunk route (problem 3). The y-axis shows the number of failed calls due to each problem on a log scale as a function of time. The most dominant problem (problem 1) drives the shape of the overall failure graph, and hides the other problems.

Persistent Problems.

Some problems, such as problem 4 from Figure 2 occur only for short durations of time, and could be discovered by change detection algorithms. However, problems such as problem 1 persisted for long periods of time, thus making it difficult to detect them via change points.

Complex triggers.

Chronics often involve only a small subset of user interactions because they are triggered by some unforeseen corner case requiring atypical conditions. An incident from the `VoIP` network illustrates this issue. Customers of a given service experienced difficulties making and receiving calls following a planned maintenance involving a configuration change to a server. The issue prevented customers whose phones used IP addresses

Table 2: Summary of Diagnosis Techniques

Technique (Canonical Example)	End-to-end tracing (Pinpoint [10])	Signature-based (Cohen2005 [6])	Graph theoretic (NetMedic [9])	Event correlation (Giza [13])
Complex problems diagnosed				
Propagating problems	✓	✓	✓	✓
Chronics	✗	✗	✗	(partial)
Multiple independent problems	✓	✗	✓	✓
Complex triggers	(partial)	✓	✗	✓

instead of fully qualified domain names from registering with the network. To effectively debug this problem, operators needed to identify that the problem occurred only for customers of the specific service when certain types of phones were used with the misconfigured server. Identifying the combination of factors necessary to trigger the problem is challenging.

3. STATE-OF-THE-ART DIAGNOSIS

Over the past decade there have been significant advances in tools that exploit statistics and machine learning to diagnose problems in distributed systems. Table 2 highlights some influential diagnosis techniques. This list is by no means exhaustive but we believe it captures the trends in diagnosis for distributed systems. This section discusses the contributions of these techniques, and their shortcomings at diagnosing chronics.

3.1 End-to-end Tracing

Some diagnostic tools [3, 5, 10, 17] analyze end-to-end request traces and localize components highly correlated with failed requests using data clustering [3, 17] or decision trees [5, 10]. They detect problems that result in changes in the causal flow of requests [10, 17], changes in request durations [17], or error codes [5]. These techniques have typically been used to diagnose infrastructural problems, such as database faults and software bugs (*e.g* infinite loops and exceptions) which lead to a marked perturbation of a subset of requests. In principle, techniques such as decision trees should fare well at diagnosing both major outages and chronics. However, decision trees did not fare well at diagnosing chronics when we applied them to our dataset. We hypothesize that the decision tree's bias towards building short trees led to the pruning of relevant features when diagnosing problems due to complex triggers. In addition, the small number of calls affected by chronics coupled with the presence of multiple independent chronics might have been mistaken for noise.

3.2 Signature-based

Signature-based diagnosis tools [4, 6, 8] allow system administrators to identify recurrent problems from a database of known problems. Research has centered on how to represent and retrieve signatures from the problem database. These techniques use Service-Level Objective (SLO) to identify periods of time where the system was behaving abnormally, and apply machine learning algorithms (*e.g.*, tree-augmented naive bayes, logistic regression) to determine which resource-usage metrics are most correlated with the anomalous periods. They generate signatures using centroids obtained by clustering feature vectors based on the resource-usage metrics correlated with the problems. These techniques can diagnose problems due to complex triggers by localizing the problem to a small set of metrics. However, they do not address multiple independent problems as they assume that a single problem occurs at a given instance of time. Chronic conditions might also go undetected by the SLOs because they are not severe enough to violate the thresholds.

3.3 Graph-theoretic

Graph-theoretic techiques analyze communication patterns across processes to track the probability that errors, or successes (*e.g.*, probes) propagate through the system. The models may also monitor violations in expected communication patterns. Graph-theoretic techniques are useful for diagnosing problems whose manifestation propagates across distributed systems.

Rish et al. [16] propose an active probing approach that drills down on the root-cause of the problem by dynamically selecting the next probe to run based on their current belief about the system state. Sherlock [2] exploits models of node behavior, such as failover mechanisms, to infer the root-cause by computing the probability that errors propagate from a set of possible root-cause nodes. NetMedic [9] uses a statistical approach that does not require extensive domain knowledge to diagnose propagating problems in enterprise systems. NetMedic diagnoses problems by capturing dependencies between components, and analyzing the joint behavior of these components in the past to estimate the likelihood of them impacting one another in the present.

These techniques can be used to detect multiple independent problems—ranking them by likelihood of occurrence. However, these techniques do not address problems due to complex triggers as they assume that the root-cause of the problem stems from a single componenent. In addition, since chronics do not severely perturb system performance they can be included in the profiles of normal behavior learned from historical data—causing chronics to go undetected.

3.4 Event correlation

Correlation can be used to automatically discover causal relationships between events in distributed systems. Oliner et al. [14] uses cross correlation to discover causal relationships between anomaly signals across components. The anomaly signals represent the changes in the behavior of components over time in terms of resource usage, message timing or semantics over time. Giza [13] exploits knowledge of the system's topology to identify spatial correlations between events, *e.g*, customers in Texas are experiencing poor video quality. Next, Giza uses cross correlation to discover causal relationships between symptoms (*e.g*, poor video quality) and diagnostic events (*e.g*, a network link is down). Cross correlation facilitates automatic rule discovery, and diagnosis of problems due to complex triggers. These techniques also support diagnosis of multiple independent problems. Giza's spatial aggregation of events can help detect chronics. However, their technique relies on correlating these chronic symptoms to diagnostic events within the service provider's network, *e.g.*, alarms—thus they can fail to localize the root-cause of the detected chronics.

4. PROPOSED APPROACH

Our overall approach consists of labeling user interactions such as phone call attempts as successful or anomalous, associating additional system-level information with these interactions using system logs, and then using a scalable ranking function to identify a group of attributes that best discriminates between the success and failure labels as our first chronic diagnosis. We then remove the interactions matching this diagnosis from the dataset and repeat the process to identify other chronic problems until a bulk of the anomalous interactions have been explained. The approach requires little domain knowledge, and does not rely on models of system behavior or historical data to localize problems. Furthermore, it is capable of localizing unanticipated problems that have never been seen before, *e.g.*, misconfiguration, failed upgrades, system overload, and unanticipated interactions due to incompatible software versions. We describe each step of the approach in more detail as follows.

4.1 Extract Call Labels and Attributes

We start from user-visible indications of failure in each individual user's interactions with the system, *i.e.*, attempts to make a phone call. Labeling of user interactions into success and failure interactions may or may not be easy depending on what information is available. For example, if logs at the user end device are available, identifying failed phone calls is easy. However, if only logs from network elements are available as in our case, domain-specific heuristics will often be required. For phone calls, a user redialing the same number immediately after disconnection, zero talk time, or server reported error code can be used as the failure indicator. In other systems, similar heuristics could work too - *e.g.*, a user repeatedly refreshing a web page, or getting a HTTP error in some part of the page. Since these labels are used for subsequent statistical analysis, occasional mislabeling can be tolerated.

We then correlate the lower-level system log data extracted from the raw CDRs with these user-level events (phone calls) to construct a "master record". The log data must have some common keys such as time, phone numbers, and IP addresses that can be used to correlate the data with the user-level event. However, the matches need not be exact, and domain-specific matching rules can be used. *E.g.*, entries may belong to the same call if the sender and receiver phone numbers match in all available digits and timestamps are within a small window of each other. Besides these keys, the remainder of each log entry is not required to have any special semantic meaning. We can treat it simply as a bag of words. For our VoIP dataset, the end result is a list of "master CDRs", one for each phone call or call attempt, and each labeled as a success or failure as shown in Figure 3.

Each master CDR consists of number of *attributes* including the calling and called phone numbers, call time and duration, a number of element names and IP addresses for elements that process the call, any defect and success codes generated by the element, the trunk lines used, and other fields present in the CDRs. Domain knowledge can be used to choose which attributes to include from the original raw logs.

4.2 Ranking Groups of Attributes

Given the set of master records, we then rank groups of attributes using a scoring function that quantifies the ability of the group to discriminate between successful calls and failed calls. To do so, we use an iterative Bayesian approach to learn a simple Bernoulli (*i.e.*, "coin toss") model of successes and failures. The idea is to model an attribute a as occurring in a call with a fixed, but unknown probability p^a. This attribute occurrence probability is p^a_f for failed calls, and p^a_s for successful calls. The model estimates these unknown probabilities using the master CDRs. However, rather than learning a single value, we can estimate the entire probability *distribution* of these unknown attribute occurrence probabilities, *i.e.*, $F^a_f(x) = P[p^a_f \leq x]$, and $F^a_s(x) = P[p^a_s \leq x]$. We start with an initial estimate for F^a_f and F^a_s, and Bayes rule is used to update this estimate as each new call in the dataset is processed, depending on whether it is a successful and failed call, and whether it contains the attribute a or not. Once these distributions are learned, the score is simply the KL di-

1. Represent call attributes as truth table

SVR1	SVR2	SVR3	PHONE1	PHONE2	OUTCOME
1	1	0	0	0	SUCCESS
0	0	1	1	0	FAIL
0	0	1	0	1	FAIL

2. Model distribution of each attribute

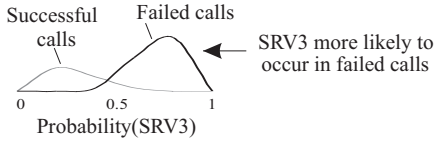

Figure 3: An overview of steps used by our top-down, statistical diagnosis algorithm.

vergence [11], a standard information-theoretic metric of the "difference" between two distributions, computed between these success and failure attribute occurrence probability distributions.

Figure 3 shows how the scoring works in terms of the density functions for the success and failure attribute occurrence probability distributions. Intuitively, it scores higher those attribute groups that are more likely to occur in failed calls than in successful calls, but it does so while taking into account the volume of data observed. This allows us to increase confidence as we observe more calls. For example, the score is higher after observing an attribute in 50 out of 100 failed calls as compared to observing it in 1 out of 2 failed calls, even though both scenarios have the same underlying probability p_f of 0.5.

We can scalably compute the score for large numbers of attribute groups and over large CDR volumes because the KL divergence can be reduced to a closed form equation due to two textbook results. The first result is that Beta distributions are *conjugate priors* for Bernoulli models, *i.e.*, if a Beta distribution $Beta(x, y)$ is used as an initial estimate for distribution F_f^a (or F_s^a), and the forward probability $P[a \text{ appears in a failed call}|F_f^a]$ (and similarly for successful calls) is given by a Bernoulli distribution, then the new estimate for F_f^a after applying Bayes rule is also a Beta distribution $Beta(x+a, y+b)$, where a and b are the number of calls with and without attribute a, respectively. The second result is that the KL divergence between two Beta distributed random variables, $X \sim Beta(a, b)$ and $Y \sim Beta(c, d)$ is given by the Equation

$$\begin{aligned} KL(Y||X) &= ln\frac{B(a,b)}{B(c,d)} - (a-c)\psi(c) - (b-d)\psi(d) \\ &\quad +(a-c+b-d)\psi(c+d) \end{aligned} \tag{1}$$

where B is the Beta function and ψ is the digamma function. Therefore, if one starts with the initial assumption that the failure and successful call attribute occurrence probabilities p_f and p_s are uniformly distributed (which is a special case of the Beta distribution), then setting a/b = 1+#successful calls with-/without attribute a, and c/d = 1+#failed calls with-/without attribute a in Equation 1 yields the desired score in Equation 2. A similar observation is used to compute KL divergences between two Bernoulli models in [12].

$$\begin{aligned} score = KL(&p(Attribute/Failure)\ ||\\ &p(Attribute/Success)) \end{aligned} \tag{2}$$

We diagnose problems involving multiple attributes by computing the score for groups of attributes. Attribute groups can be produced systematically, by using an inverted index to search for attribute groups having a large number of successes and failures, they can be produced by random sampling, or they can be produced by using domain-specific heuristics. Currently, we use a inverted index to produce attribute combinations systematically.

After picking the group with the highest score as the first problem to be discovered, we then remove all calls (both success and failures) that match this attribute combination from the dataset, and then repeat the process. Doing so removes the impact of the first diagnosed problem and allows us to ask what explains the remaining failures. In this manner, we can identify separate independent failure causes.

4.3 Preliminary Successes

Application of our approach to VoIP data logs has led to very promising results. We analyze 25 million records in 15 minutes (*i.e*, 7 minutes to load the data + 8 minutes of diagnosis time) on a 8-core Xeon HT (@2.4GHz) with 24GB of memory. We have been able to help the chronics team quickly identify several new problems. We list a few such instances below.

Incident 1. A repeating increase in the number of defects during night hours was observed associated with a given defect code illustrated in Figure 4. Our analysis identified two different (business) customers as being associated with the bulk of the defects. While these customers accounted for large share of total defects, the defect rate observed by the customers were a fraction of one percent. The operations team determined that these two customers were attempting to send faxes overseas using unsupported codecs during US night time.

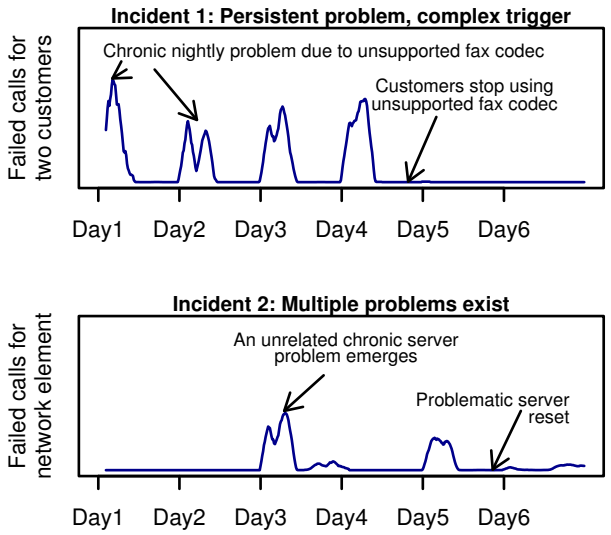

Figure 4: Multiple chronic problems exist at a production system.

Shortly after the date the customers were notified of the problem, the daily defect count associated with this defect code decreased by 56%.

Incident 2. Our analysis identified an independent problem with a specific network element that occurred concurrently with incident 1 (see Figure 4), and accounted for over 50% of the remaining defects when failures due to Incident 1 were excluded. Again, overall only a fraction of one percent of the calls passing through this element were failing making the problem harder to identify. After the operations team reset the element, the total number of daily defects associated with this defect code was reduced by 76%, and this element was no longer implicated by our analysis.

Incident 3. An increase in failure rate during business hours was observed for a single defect code (see Figure 5). Our analysis identified a trunk group as being associated with up to 80% of these defects. At peak, 2-3% of the calls passing this trunk group would fail. Analysis by the operations team revealed two blocked CICs (Circuit Identification Codes) on the trunk group and as a result the problem would only affect calls assigned to these blocked CICs (in a round robin manner). After those CICs were unblocked, the total defects associated with this code were reduced by 80%.

4.4 Why does it work?

A number of the characteristics of our approach allows it to deal well with the challenges introduced by chronic defects.

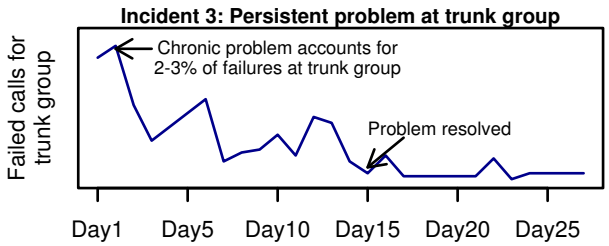

Figure 5: Chronic problem affects 2-3% of calls at trunk group in production system.

1. Separation of concurrent root causes. Our iterative analysis allows us to separate and localize the different causes of many concurrent chronics that persist at any given time, even if they share attributes (*e.g.*, pass through the same network element). By repeatedly filtering out calls that match dominant problems as they are detected, we expose increasingly smaller problems that may earlier have been hidden in the noise.

2. Identification of novel problems. An undiagnosed chronic problem is often novel, *i.e.*, something that operators have not seen before. For example, it may involve a new network element, or a new customer (with potentially novel configuration issues). Since we do not rely on models learned from "normal" operation or signatures of known defects, our approach can be used for problems that have never been seen before.

3. Identification of low-grade problems. Threshold-based approaches often miss chronics because of their low numbers. We forego thresholds by using comparisons between successful and failed calls to identify attributes discriminative of failures. The Bayesian inference we use can update the success and failure distributions with very few calls. Therefore, our approach can detect problems with very small numbers of failures.

4. Identification of persistent problems. In some cases, chronic problems cause failed user requests consistently over time or consistently with the workload in the system. Any approach that attempts to identify differences from "normal" behavior to identify sudden changes or spikes will fail to detect them (today is no different from yesterday), but since our approach does not restrict us to time as the only discriminative element, it is able to diagnose such problems.

5. Identification of complex triggers. Because our approach evaluates many groups of attributes against the scoring function, it can identify problems that occur only when multiple conditions (encoded by attributes) are satisfied at once.

4.5 Next Steps

Our approach requires traces that label every user-

interaction with the system as successful or anomalous. The focus of our approach is not anomaly-detection but rather localizing the root-cause of problems once anomalies are identified. At present, our approach assumes the call attributes are binary, and does not cater for real-valued attributes such as CPU and memory-usage. We have observed false-positives when the underlying root-cause is not in the call logs, *e.g.*, router failures. In these instances, we implicate network elements adjacent to the faulty router. We are extending our approach to incorporate additional data sources such as router logs, and to cope with real-valued data.

5. CONCLUSION

This paper discusses the challenges associated with discovering and diagnosing chronics, *i.e.*, recurrent problems that fly under the radar and do not trigger alarm thresholds. We present a new statistical approach for the diagnosis of never-before seen chronics that does not rely on models of system behavior, or historical data to localize problems. Our approach uses a scalable Bayesian distribution learner coupled with an information-theoretic measure of distance, to identify the sets of attributes that best distinguish failed requests from successful requests. We present preliminary results which demonstrate the usefulness of our approach for diagnosing actual chronics in a large `VoIP` system. Despite our focus on `VoIP` systems, we believe that our findings can be applied to other large-scale distributed systems that log the outcomes of individual user transactions.

6. REFERENCES

[1] M. K. Agarwal, M. Gupta, V. Mann, N. Sachindran, N. Anerousis, and L. B. Mummert. Problem determination in enterprise middleware systems using change point correlation of time series data. In *NOMS*, pages 471–482, Vancouver, Canada, April 2006.

[2] P. Bahl, R. Chandra, A. G. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *SIGCOMM*, pages 13–24, Kyoto, Japan, August 2007.

[3] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *OSDI*, pages 259–272, San Francisco, CA, December 2004.

[4] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *EuroSys*, pages 111–124, Paris, France, April 2010.

[5] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *ICAC*, pages 36–43, New York, NY, May 2004.

[6] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. In *SOSP*, pages 105–118, Brighton, United Kingdom, October 2005.

[7] Comcast. The Comcast Corporation, 2011. `http://www.comcast.com`.

[8] S. Duan and S. Babu. Guided problem diagnosis through active learning. In *ICAC*, pages 45–54, Chicago, IL, June 2008.

[9] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *SIGCOMM*, pages 243–254, Barcelona, Spain, August 2009.

[10] E. Kiciman and A. Fox. Detecting application-level failures in component-based internet services. *IEEE Trans. on Neural Networks: Special Issue on Adaptive Learning Systems in Communication Networks*, 16(5):1027–1041, September 2005.

[11] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, March 1951.

[12] C. Liu, Z. Lian, and J. Han. How Bayesians debug. In *ICDM*, pages 382–393, Hong Kong, China, December 2006.

[13] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large IPTV network. In *SIGCOMM*, pages 231–242, Barcelona, Spain, August 2009.

[14] A. J. Oliner, A. V. Kulkarni, and A. Aiken. Using correlated surprise to infer shared influence. In *DSN*, pages 191–200, Chicago, IL, July 2010.

[15] Qwest. Qwest Communications International, Inc., 2011. `http://www.qwest.com`.

[16] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik. Real-time problem determination in distributed systems using active probing. In *NOMS*, pages 133–146, Seoul, South Korea, April 2004.

[17] R. R. Sambasivan, A. X. Zheng, M. D. Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. R. Ganger. Diagnosing performance changes by comparing request flows. In *NSDI*, pages 43–56, Boston, MA, March 2011.

[18] M. Sauter. *Beyond 3G - Bringing Networks, Terminals and the Web Together: LTE, WiMAX, IMS, 4G Devices and the Mobile Web 2.0*. Wiley Publishing, 2009.

[19] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *Communications Magazine, IEEE*, 34(5):82–90, May 1996.